DARPA Urban Challenge
May 31, 2007
Insight Racing Team

Grayson Randall
Team Lead and Author
grayson@insightracing.org

Authors:
Amit Bhatia
Brian Deery
Steve Kuekes
Mary Ellen Randall
Michael Randall
David Scott

1.0 EXECUTIVE SUMMARY **1**

2.0 INTRODUCTION AND OVERVIEW **1**

2.1 OVERALL PROBLEM TO BE SOLVED 1
2.2 ISSUES THAT DRIVE DESIGN CHOICES 2
2.3 DESIGN APPROACH 2
2.4 HIGH LEVEL ARCHITECTURE 3
2.4.1 Object Sensing Systems *4*
2.4.2 Location Sensors *6*
2.4.3 Route Analysis and Management *6*
2.4.4 Vehicle Monitoring *7*
2.4.5 Scene Analysis Module (SAM) *7*
2.4.6 Path Planning Module (PPM) *7*
2.4.7 Vehicle Control Subsystem *7*
2.4.8 Mission Objective Module (MOM) (..... *7*
2.4.9 System Communication *8*
2.4.10 Power Systems for the Vehicle *8*
2.4.11 E-Stop System *10*

3.0 ANALYSIS AND DESIGN **10**

3.1 PATH PLANNING MODULE (PPM) 10
3.1.1 Design Choices *10*
3.1.2 PPM Results & Performance *12*
3.2 ROUTE ANALYSIS & MANAGEMENT (RAM) ANALYSIS & DESIGN 12
3.2.1 Overview *12*
3.2.2 Route Solving Design Choices *13*
3.2.3 RAM Algorithm Performance *13*
3.2.4 Current State / Next State Prediction Design Choices *14*
3.2.5 Intersection Identification and Management *14*
3.2.6 Fault Tolerance *15*
3.3 CAMERA PROCESSING 16
3.3.1 Design Choices *16*
3.3.2 Camera Results *17*
3.3.3 Observations and Analysis *18*
3.4 LIDAR PROCESSING 19
3.4.1 Analysis and Design *19*
3.4.2 Lidar Processing Results and Performance *20*
3.5 TESTING 21
3.5.1 Methodology *21*
3.5.2 Simulation *21*

4.0 TEST STATUS **22**

5.0 SUMMARY **22**

6.0 REFERENCES **23**

1.0 Executive Summary

Insight Racing's design is based on three "Golden Rules" in the following order:

1) No Collisions
2) Follow All Traffic Laws
3) Complete Every Mission

Key Findings and novel approaches include:

- Development of a modular architecture to integrate large amounts of sensor data in a real time environment
- A combining of techniques to achieve real time processing of camera images scanning the captured color image for yellow and white regions and extracting lines used in lane marking and parking lots
- Miniaturization of the solution to meet vehicle space constraints of a sports car, with special considerations for power and cooling
- Mathematical techniques for identification and combining objects, both moving and static
- Efficient route planning between checkpoints
- Efficient path planning between obstacles

In utilizing the architecture and system described herein, Insight Racing's Urban Challenge entry is running at speeds up to 25 miles per hour and averaging approximately 18 mph on a mission. At the time of this writing, testing of behaviors known as Advanced Navigation is underway.

2.0 Introduction and Overview

2.1 Overall Problem To Be Solved

The Urban Challenge is an autonomous vehicle race where full sized cars and trucks compete in an Urban Setting. These vehicles drive around the city unassisted. They interact with each other, follow traffic laws, navigate intersections, traffic, park, and pass other vehicles.

The following capabilities are needed in our Urban Challenge entry: lidar processing for obstacle detection and avoidance, master control module for decision making, control of brakes, throttle, transmission, steering, capability to operate in reverse, high speed image processing and radar for longer range obstacle detection and avoidance, position analysis, sparse waypoint operation, operation up to 30 mph, operation without Global Positioning System (GPS) signal, sensing of road edges, and driver display and monitoring module.

2.2 Issues That Drive Design Choices

The new design challenges for the Urban Challenge are:

- Vehicle routing algorithms
- Path planning
- The need to collect data from the 360° area around the vehicle
- Detecting lines which denote road and parking lot markings
- Integration of multiple sensors with significant data processing requirements
- Awareness of position and operation of other moving vehicles
- Operating at intersections
- Operation in an unstructured environment such as a zone or parking lot
- Real time processing of camera images to support 30+ mile per hour speeds
- The need to isolate various design elements for testing

Insight Racing is using a Lotus Elise sports car as its Urban Challenge entrant which imposes additional vehicle space constraints over some other Urban Challenge teams.

2.3 Design Approach

This design approach is based on the three (3) "Golden Rules in the following order:"

1) No Collisions
2) Follow All Traffic Laws
3) Complete Every Mission

There are several reasons why Insight Racing designed this System Architecture.

1) It is an architected system (See Figure 1.). The architecture allows for a modular design. These design units can be developed and tested individually and then integrated for final test. Any issues in the system can be easily isolated to architected interfaces. The ability to add new sensors and to remove sensors is very easy with an architected modular design.

2) The architecture is based on a distributed processing model. This allows us the ability to run unique code modules on physically different processors and allows us the ability to run any specific code module on multiple physical processors to support redundancy of key modules. These features allow us the ability to easily balance the load on the processors and to use excess processing power to run redundant tasks.

3) Because we use an architected modular design and distributed processor model, we are able to easily scale our overall control system to any size platform. It can be scaled to add many sensors for a large system or to remove sensors to create a very simple system. All the physical I/O is confined to a single design module. This allows us to replace the physical control system without impacting any of the sensors processing, path planning, or decision making modules.

Insight is using a Lotus Elise to provide a miniaturized solution, suitable to both military and commercial use. Insight is mounting equipment in the trunk, on a platform mounted on top of the Elise, and on special front and rear bumpers. Sensors will be mounted on the front, rear and top of the Elise. Some of the top sensors are mounted to rotate to look left and right of the vehicle. The trunk is fitted with computers, power, additional cooling and networking equipment.

Insight will use 9 small format mini computers running Linux for processing power with Gigabit Ethernet interfaces. All the computers will be running Linux with a C based development environment. We will use Ethernet to access our sensor data which is concentrated in Comtrol units stored in the trunk. The Comtrol devices concentrate either serial devices or SICK high speed Lidar data into a single device that can be accessed on the Ethernet. Each processor can access an individual device with a Transmission control Protocol/Internet Protocol (TCP/IP) socket, allowing us to move modules to different processors and to fail-over processes without physically moving any sensor's physical connection. This worked very well in our 2005 Grand Challenge vehicle. We will use a similar architecture to our previous vehicle, enhanced to work with more processing power and more sensor data. Some of our video modules will use C with assembler subroutines to allow access to the processor SIMD (Single Instruction Multiple Data) instructions.

2.4 High Level Architecture

Our strategy to balance load on the computers is to distribute processing as far to the outside edge of the architecture as possible (reference figure 1). Then we will move the object description blocks to the center where we make decisions based on all the available information.

Our reliability strategy includes many facets. First, our sensors overlap in coverage and technology. We overlap key areas with radar, cameras and lidars to ensure we identify all objects. The sensor inputs are handled using unique processing modules and then compared for consistency and completeness. This is important based on the significant noise in the sensor systems.

In addition, one specialized module monitors all the processes and re-initializes processes, if needed. This monitoring module also redistributes processing to other physical processors in the case of hardware failure. Key processes will be run on multiple physical processors. This monitoring module is based on technology developed for our 2005 Grand Challenge entry.

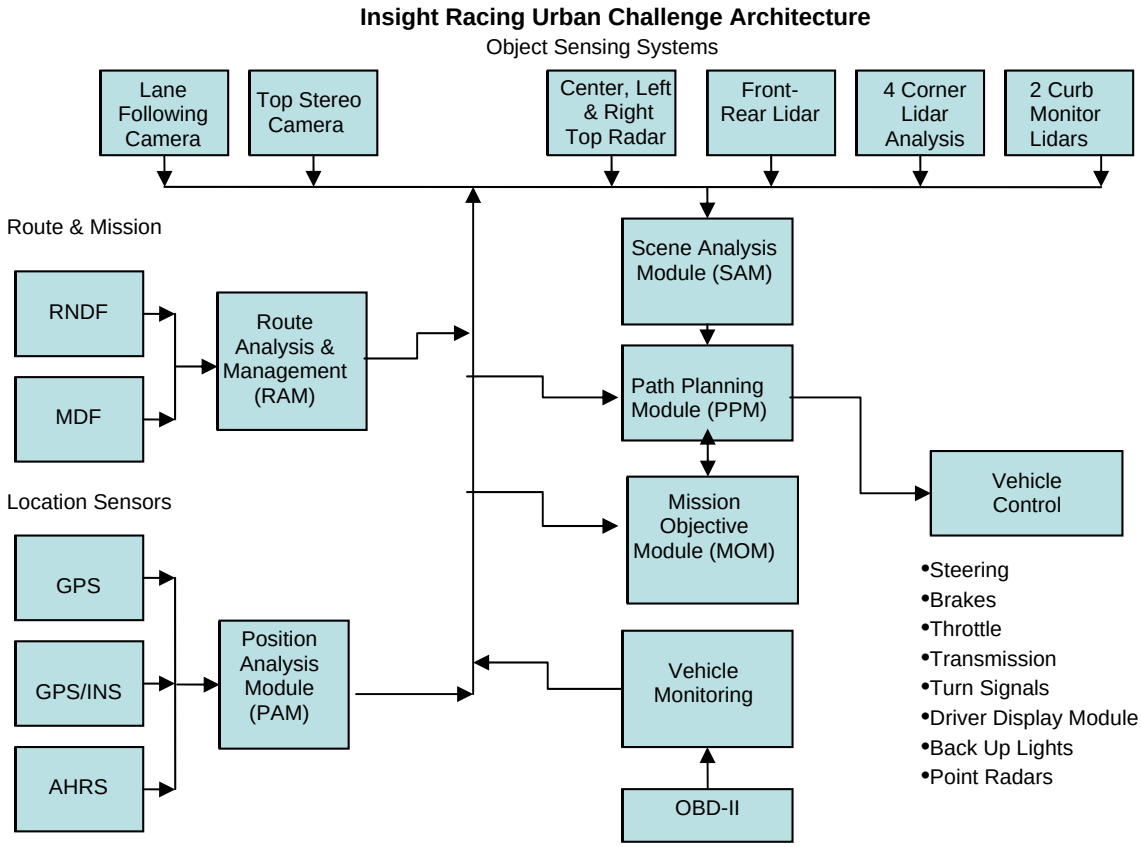

Figure 1 - Urban Challenge Architecture

2.4.1 Object Sensing Systems

Each sensor will process raw sensor data and identify objects within its field of view. The Object Sensing modules include processing for: Front & rear Lidar units, 4 Corner Lidar units, Lane following camera, Top Stereo Camera, and 3 radar units (See figure 2).

All the object information from each of the object sensing subsystems is collected by the Scene Analysis Module (SAM). SAM will fuse all the object information and form a single 360 degree map around the vehicle. Each of the objects will be combined based on overlap between the sensors and will be represented based on the object's direction and speed. This map will then be used for path analysis to determine correct direction of the Urban Challenge vehicle at that point in time. This will include obstacle avoidance. The SAM map is based on an x,y meter grid. All lat/long positions are transferred to this flat space. All objects are referenced to a specific x,y location which helps maintain correct positions during turns and changes in speed.

Sensor Placement Aerial View

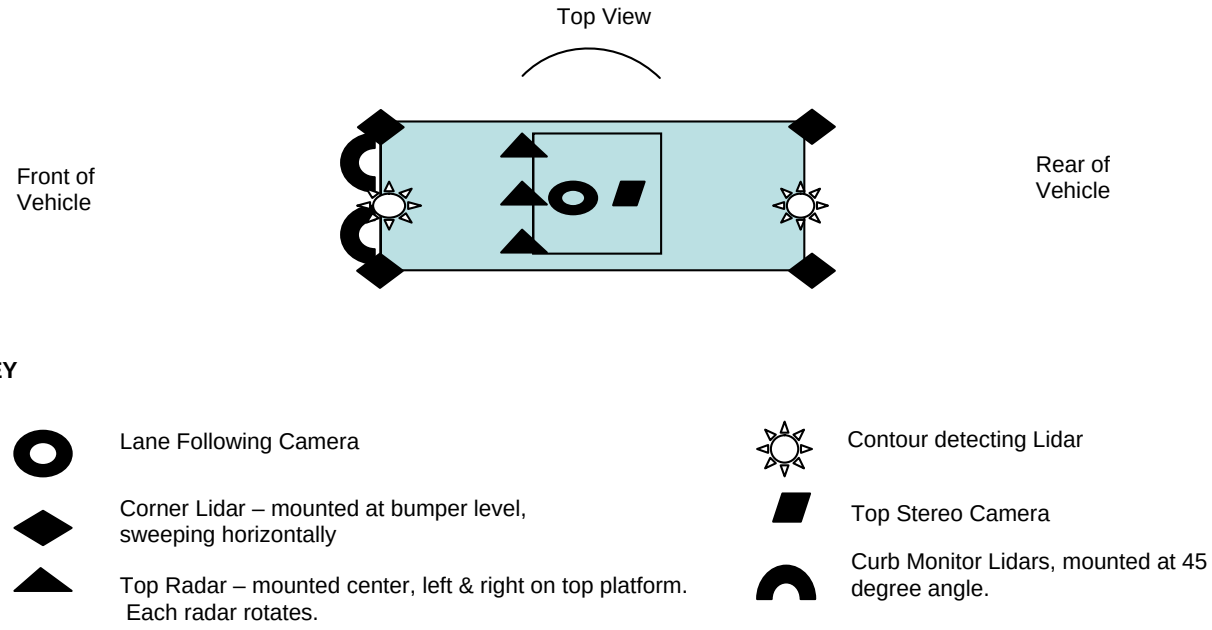

Figure 2 - Object Sensing System

2.4.1.1 Lane Following Camera - To determine the lane markings and/or the edge of the road on the left and right side of the vehicle. This information is used to center the vehicle in the lane. The camera is located in a fixed position elevated above the vehicle roof platform near the front. It looks down approximately 45 degrees from the horizontal to get good visual identification of the lane. (See Figure 2)

2.4.1.2 Front & Rear Lidar - To determine the contour of the terrain in front and back of the vehicle. It is also used as a secondary determination of other vehicles in the same lane. Lidar is mounted on the front and rear bumper near the center. Lidar scans in a vertical plane. (See Figure 2)

2.4.1.3 Curb Finding Lidars – To determine the location of features that may identify the edge of the road. This will include curbs, ditches, medians and other objects that have a vertical profile with respect to the road surface. These lidars are mounted at a 45° angle on the front bumper just inside each corner and have a vertical sweep. This allows the vehicle to see the road surface and any vertical projections that it may be approaching. This will also provide useful information when on dirt roads as it will help determine features at the edge of the unmarked road. (See Figure 2)

2.4.1.4 Top Stereo Camera - To determine range information (distance) to other vehicles and objects in front of our vehicle. This is mounted on the top platform facing forward. This is similar to the mono cameras in that it looks forward during driving. (See Figure 2)

2.4.1.5 Left & Right & Center Top Radar - To determine the objects around the vehicle. Three automotive radar units will be placed on the vehicle. One will be placed at the center of the top platform. The second will be placed at the left top of the platform. The third unit will be placed at the right top platform. Each radar may be moved 180 degrees and will help detect oncoming traffic at intersections. They will also be used to detect longer range objects. The Delphi radar has the capability to capture static and dynamic objects at a distance of up to 150 meters. (See Figure 2)

2.4.1.6 Four Corner Lidar - To determine the precise location of vehicles and objects in close proximity to our vehicle. The lidars are placed in each of the four corners of the vehicle at approximately the bumper level. They are mounted on the front and rear bumpers and sweep a horizontal plane parallel to the ground. These devices are key to maintaining vehicle spacing and to support all traffic merging operations. These devices will also support object detection at slower speeds. (See Figure 2)

2.4.2 Location Sensors (See Figure 2)
The location sensors will determine our position, heading and speed based on a variety of sensor inputs. These are both redundant and independent. The Position Analysis Module (PAM) includes processing for: GPS, GPS/INS (Inertial Navigation System), and Attitude Heading Reference System (AHRS) units. PAM will output a single position, heading, and speed based on all input.

2.4.2.1 GPS - To determine the location, direction and speed of the vehicle. The GPS has 10 centimeter accuracy using Omnistar HP service. This device is used as a back up for position.

2.4.2.2 GPS/INS - To determine the location, direction and speed of the vehicle. This device uses inertial navigation to allow us to maintain position information when GPS signals are not available. The device synchronizes the GPS with the INS automatically and maintains accurate position information, even when GPS signals may not be available.

2.4.2.3 AHRS - To determine the attitude and direction of the vehicle. This sensor gives back magnetic heading information as well as 3 axis acceleration and positions, representing the attitude of the vehicle. Yaw rate is used as input to the radar systems. This is back up to the INS.

2.4.2.4 OBD-II - To collect distance and speed information from the vehicle OBD-II (On Board Diagnostics) interface. This is used to determine position based on dead reckoning. It is an imprecise system, but allows us a back up to the more accurate systems. We can also access wheel rotation information from the OBD-II interface as additional input.

2.4.3 Route Analysis and Management (See Figure 1)
The route and mission processing module will use the DARPA defined RNDF (Route

Network Data File) and MDF (Mission Data File) as its input. It will determine routing and will provide the route information to the Path Planning Module (PPM) and to the Mission Objective Module (MOM). Based on current position, learned object positions, and mission objectives, this module will use routing algorithms to calculate the fastest route to the next checkpoint. It will also reroute the vehicle in the event a route is blocked. This module will collect and remember information about the location of static objects along the course and in zones.

2.4.4 Vehicle Monitoring (See Figure 1)

Vehicle Monitoring will collect data about the current operation of the vehicle. Most of this data will be obtained from the On Board Computers, using the On Board Diagnostics (OBD-II) interface. Data includes Revolutions per Minute (RPM), engine temperature, wheel rotations, fuel levels, and other related vehicle information.

2.4.5 Scene Analysis Module (SAM) (See Figure 1)

The SAM collects the data from all the object sensing subsystems. SAM is responsible to fuse all the incoming data into a 360° map of the area around the vehicle. SAM resolves all duplicate objects from the multiple sensors and projects objects that may be temporarily out of view. This information will be used to increase the aggressiveness of the vehicle as it becomes more aware of its surroundings and to reduce the volume of object data that gets processed.

2.4.6 Path Planning Module (PPM) (See Figure 1)

This module takes the scene map from SAM, the state information from MOM, and the route information from RAM (Route Analysis and Management). PPM projects the position and trajectory of all moving objects and calculates an appropriate path and speed for our Urban Challenge vehicle to proceed safely. It will feed back to MOM information required by the state machine concerning the location and movement of objects relative to our Urban Challenge vehicle and sends speed and direction (heading) commands to the vehicle control system.

2.4.7 Vehicle Control Subsystem (See Figure 1)

The Vehicle Control Subsystem is where the physical input/output is performed with the vehicle. The vehicle systems controlled include: Steering, Brakes, Accelerator, Transmission Shift, Blinkers, Rotation of Top Radar Units, Driver Display Module (used for debug), and Back up Lights. Primary inputs to this subsystem are speed and steering angle, as well as inputs to other devices. This module has no awareness of its surroundings outside the vehicle. This is the only module that has code specific to the vehicle in which it is operating.

2.4.8 Mission Objective Module (MOM) (See Figure 1)

This module is responsible to make all decisions required to execute and complete the current mission. This is implemented as a rules-based state machine. Other modules in the system will make specific decisions based on what state the system is in. The primary states are included in table 1.

This is a preliminary list of valid states (See Table 1). This list will be updated as required during development and testing

The MOM state machine utilizes states from Table below:

State Definition		State Definition		State Definition
Lane Following		Turn Left		E-stop Pause
Passing		Turn Right		E-stop Disable
Shift Lane Left		Turn Center		Panic Stop
Shift Lane Right		Zone Find Parking		Stop
Intersection Approach Stop		Zone Park		Blocked
Intersection Approach Rolling		Zone Back Out		U-Turn
Taxi Cab Rules		Zone Find Exit		Reverse

Table 1 – Primary States for State Machine

2.4.9 System Communication

The communications manager is based on the Jaus (Joint Architecture for Unmanned Systems) standard [1] for unmanned vehicle communications. It is a message passing protocol used for communication between all the sensors and major architectural units in Insight's system.

2.4.10 Power Systems for the Vehicle

The power system is based around a 24v DC automotive design. The stock 12v alternator was replaced with a larger 24v model that has enough capacity to power all the systems. The power distribution diagram is shown in figure 3.

The alternator's voltage regulator is equipped with self-monitoring circuitry. It can send out via Ethernet data such as line voltage and load utilization. The computer programs can determine over the network if the alternator is being taxed at full load, such as when the batteries are being charged. The software can also determine if the alternator is malfunctioning, such as when the output voltage is too high or low.

Operating both 12v and 24v systems off only 2 batteries would cause uneven discharge between each battery. One battery would get deeply discharged while the other would become severely overcharged. This problem is mitigated by the addition of a battery equalizer, which balances the load between the batteries. The batteries will both discharge and charge at the same rate. They will have a much longer useful lifespan and makes using 12v devices in a 24v system practical.

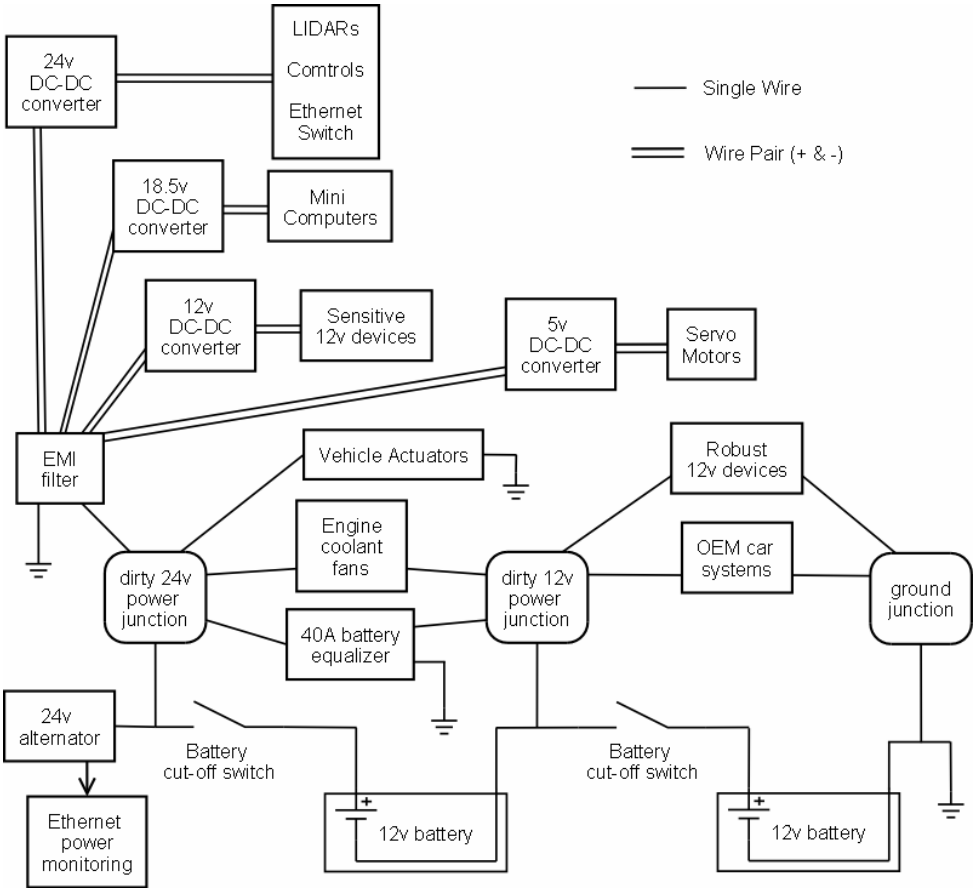

Figure 3 – Power Distribution Diagram

Two battery cut-off switches, when both switched off, disconnect all power from the vehicle. This is the primary means to power down the vehicle for the night. The batteries also regulate the alternator voltage, so they must not be disconnected while the engine is running.

Since the original vehicle runs on 12v, the original systems were connected across one of the two batteries. To balance the load between the batteries, some 12v equipment draws power from the 2nd battery. The battery equalizer compensates for the difference in power usage between the batteries.

The alternator generally produces 26v or so to keep the batteries charged. It also has the potential to produce electrically noisy power (dirty power). Sensitive computer equipment does not operate well with this kind of power. To ensure reliable power for the sensitive devices, power is first passed though an EMI filter to remove some of the high frequency noise. DC-to-DC voltage power supplies take in the dirty power and produces clean power at voltages required by the various devices. For example, 26v, the normal operating voltage of a 24v system, is above the safe operating voltage range for the SICK lidars. Any errant voltage spikes from the alternator could permanently disable the lidars if they were connected to dirty 24v power.

DC-to-DC power supplies are also used to power all the mini computers. The power supplies are substitutes for the power bricks they were sold with. DC-to-DC supplies power all the other computerized equipment. Exceptions are made for equipment that is designed to be powered by an automobile, such as the Radar units. Those can be powered using unfiltered power from the alternator, as they would be in a production vehicle.

2.4.11 E-Stop System

Our E-Stop system uses multiple E-Stop switches located on the exterior of the vehicle and in the passenger compartment. These are designed to be fail-safe and to activate if any connection is broken. This design will stop the vehicle if any E-Stop wire becomes broken, the heartbeat signal from the software is lost, the signal from a remote E-Stop controller is lost, or a if commanded from the remote E-Stop controller. We support either our Insight designed remote E-Stop controller or the Omnitech Robotics E-Stop controller used by DARPA.

3.0 Analysis and Design

The key areas where complex mathematical techniques were needed are: PPM, RAM, Camera Processing, and Object Detection.

3.1 Path Planning Module (PPM)

3.1.1 Design Choices

The Path Planning Module receives input from the RAM, MOM, PAM, SAM, and Vehicle Control subsystems. PPM checks the route for obstructions, finds a clear path, and finds shortest path.

PPM creates a list of waypoints to drive using input from the waypoint list, curb finder, and lane finder. The distance ahead that PPM looks is proportional to the speed the vehicle is currently or planned to travel. It then checks that route for obstructions using vector analysis. Object polygons are expanded by ½ of the vehicle width plus a safety margin using Minkowski Sums [2], [3].

Minkowski sums are the sum of two polygons. For this application one of the polygons is the vehicle and the other polygon is an object. The center of the vehicle polygon is added to the object polygon effectively growing it by ½ the vehicle width.
See Figure 4.

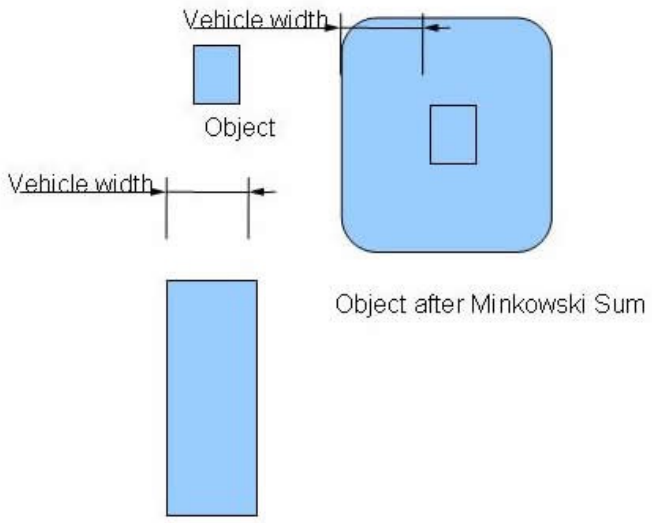

Figure 4 - Application of Minkowski Sum

After summing all the objects around the vehicle, the vehicle can now be represented by a line. If the line can traverse the desired path without intersecting any of the expanded objects, then there is room for the vehicle to operate on that path. If the line intersects any of the objects then a new path is computed. As you can see in the following example, the direct route from the vehicle location to the waypoint destination intersects with the expanded polygons. Therefore the direct route would collide the normal vehicle polygon with the normal objects. See figure 5 which follows.

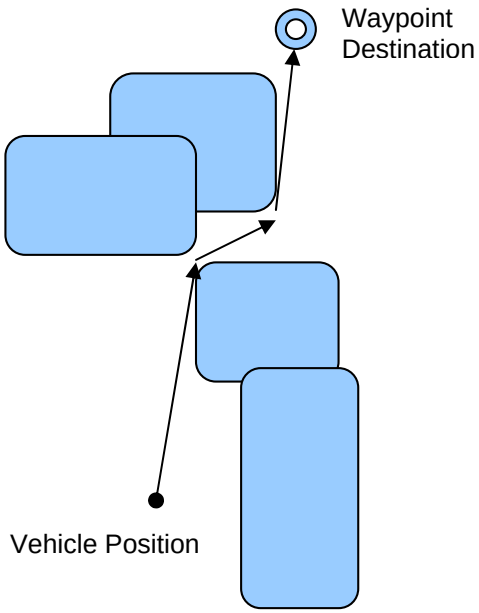

Figure 5 - Route through Minkowski Summed Objects

The new path around the objects is determined by creating a visibility graph [4] for each object corner, the current vehicle position, and the ending waypoint. That graph is traversed using a Dijkstra's [5],[6] shortest path algorithm. The new path is then used as the vehicle proposed path.

3.1.2 PPM Results & Performance

This combination of Minkowski sums, visibility graphs, and Dijkstra's shortest path algorithms provides an exact way to ensure that the vehicle does not impact any objects and takes the shortest path around objects to reach waypoint destinations.

If there is no route around the object by staying in the current travel lane, then blockage information is sent to the MOM module for decision processing and the vehicle will stop a safe distance from the object. MOM can then choose to have the vehicle shift lanes, wait for the blockage to clear, or make a U-turn.

Once a clear path is established then PPM calculates the maximum speed the vehicle can travel. The radius of any turns in the proposed route is calculated and the vehicle speed is limited according to the Centripetal force in the corners. For cornering the vehicle will not exceed a maximum lateral G force. The G force is used to calculate the maximum speed that the vehicle can travel.

V=sqrt(fr)

where f is the maximum force, v is the velocity maximum and r is the radius of the turn.

The PPM module is dependant on timely input information from other components to accurately drive the vehicle without contacting objects. If any required information is missing then the PPM module will pause the vehicle and wait for that component to be automatically restarted, before continuing.

3.2 Route Analysis & Management (RAM) Analysis & Design

3.2.1 Overview

The Route Analysis & Management module (RAM) performs the following functions:

1. Process the Route Network Data File (RNDF) and maintain an in-memory route network
2. Process the Mission Data File (MDF) and maintain a list of checkpoints for the mission
3. Find routes between each of the checkpoints
4. Update the list of completed waypoints/checkpoints and broadcast the list of next waypoints along the mission
5. Broadcast information about stop signs, turns, curves and intersections

6. Broadcast a recommended state for the vehicle based on current position, speed heading and route and mission information
7. Anticipate and recommend the next state for the vehicle
8. Receive information about static road blocks and re-plan the mission as necessary

3.2.2 Route Solving Design Choices

3.2.2.1 Processing the RNDF

The first function required of the RAM module is to process the RNDF. The RNDF is a text file containing a description of the network of roads where the autonomous mission is to take place. A complete description of the RNDF syntax is found in [7]. RAM parses the RNDF and builds an internal data structure representing the road segments, lanes, checkpoints, waypoints and other pertinent data.

3.2.2.2 Processing the MDF

Next the MDF is read in and processed (also defined in [7]). The MDF contains a list of checkpoints that must be visited to complete the mission. RAM identifies the list of waypoints that make up the shortest route between each of the consecutive checkpoint pairs. A greedy algorithm is used to quickly find a "good" route. Time is used as the cost function for traveling from waypoint to another, and the next waypoint chosen is the one that is closest (using a straight-line distance) to the destination checkpoint. Stop signs incur an additional time penalty, as do traveling through parking lots, to discourage planning a route through them unless necessary.

Once an initial route is found, the time required to traverse the route is used as the upper limit. A binary search is then conducted by searching for another route that can be completed in ½ the time, using the same greedy algorithm. While searching for a route, if the route's time exceeds the specified time limit, the route is abandoned and the algorithm backtracks to the next greediest waypoint.

If a route is found within the time limit, the time is reduced (by 1/2) and the search is repeated. If not found, the time is increased (by ½) and the search is repeated again.

Once the list of routes required to complete the mission has been found, a subset of the list of waypoints is broadcast to other components via Jaus Manager for defining the primary driving path.

3.2.3 RAM Algorithm Performance

As expected, the performance of the RAM algorithm depends primarily on the size of the RNDF, but is basically linear with respect to the number of checkpoints (refer to Table

2). For example, an MDF containing 1,000 checkpoints was processed using the Sample RNDF provided by DARPA [15]. Approximately 1.9 seconds were required to determine all 999 routes. In a second example, 10,000 checkpoints (9,999 routes) were processed in less than 20 seconds. This latter example also demonstrates that the RAM module is robust enough the handle large volumes of data.

Number of Routes (Checkpoints - 1)	Total Time (sec)	Average Time Per Route (sec)
9	0.015	0.001667
999	1.938	0.001940
9999	19.718	0.001972

Table 2 – Checkpoint Processing Time

3.2.4 Current State / Next State Prediction Design Choices

Once the waypoint list has been identified, RAM begins monitoring the vehicle's position based on position, speed and heading information provided by the Path Planning Module (PPM). Using the RNDF information and position, RAM computes a recommended state for the vehicle's operation. The recommended state is comprised of a primary state and a sub-state. The primary state identifies a general disposition for the vehicle (normal driving, in an intersection, in a parking lot, etc.) The sub-state provides additional information about the primary state (following in a lane, turning, in a curve, finding a parking spot, stopping, etc.) This state information is used to signal other modules about the various driving maneuvers required (turning, parking, etc.). Based purely on position and defined routes, this module is not aware of an influences outside of the vehicle.

It is not enough to know the current state of the vehicle (approaching a stop line for instance.) The RAM module also predicts the next state for the vehicle and broadcasts it for other modules to use. For instance, in an intersection, it is important to know that the vehicle will be turning left, after it has stopped at the stop line, so that the turn signal can be turned on prior to the turn.

The RAM module provides additional information on upcoming intersections, turns, curves, stop lines and, u-turns. The distance to these points is broadcast, as well as the angle for turns and curves. This information helps the driving modules make appropriate changes in speed and direction to negotiate these points correctly.

3.2.5 Intersection Identification and Management

When the vehicle approaches an intersection, the RAM module provides information about the structure of the intersection. Lanes where other traffic may interfere with the vehicle's progress are identified, and a relative heading to waypoints in these lanes is broadcast. This information allows the various sensors (lidars, radars, etc) to know where to scan the intersection to identify other traffic.

3.2.6 Fault Tolerance

Like all software modules in the Lone Wolf, the RAM module must be fault tolerant. It supports:

- Checkpoint/Restart capability
- Waypoint look-ahead capability
- Automatic route re-planning
- Checkpoint bypassing

3.2.6.1 Checkpoint/Restart Capability

As each checkpoint is completed, RAM writes certain mission related information to disk. In the event of a hardware failure, software failure, or other problem before a mission is complete, the RAM module may be restarted by the controlling software. When restarted, RAM will restore the current mission related information, re-plan routes where necessary based on the vehicle's current position and resume the process of broadcasting waypoints, recommended states and other mission information to other modules.

3.2.6.2 Waypoint Look-Ahead Capability

During the course of a mission, the vehicle may be traveling at a relatively high rate of speed as compared to the proximity of some number of waypoints. If waypoints are placed relatively close together (as in the case of a curve or a turn for instance), it may be possible for the vehicle to bypass a waypoint before the RAM module has determined that the criteria for visiting the waypoint have been satisfied. Or, due to obstacles in the path to a waypoint (or at the waypoint itself), it may not be possible for the vehicle to get close enough to the waypoint to consider it visited.

To ensure that the RAM module does not get stuck expecting the vehicle to visit a waypoint that has already been visited by the vehicle, the RAM module "looks" ahead some number of waypoints, and makes decisions about visited waypoints based on the vehicle's position compared to several waypoints.

3.2.6.3 Route Re-planning

Ordinarily, once the mission route has been planned, it should not have to be re-planned again unless the vehicle encounters some sort of a road-block that prevents passage. Periodically however, the RAM module compares its current position to the expected position along the route. If too much of a discrepancy exists between the current position and expected position, the RAM module automatically re-plans the route from its current position to the next checkpoint.

In order to re-plan successfully, the RAM module must be able to locate the "best" waypoint that is relatively close to its current position to re-establish its location in the route network. This feature is needed in cases where some failure causes the RAM module to be restarted, or in the event that some other condition causes the vehicle to deviate from the waypoint list being broadcast by the RAM module. By re-planning only the affected portions of the route, time is saved during the re-planning process. This can be crucial for instance, if some evasive action is required by the vehicle.

3.2.6.4 Checkpoint Bypassing

In the event that the RNDF has an error or a valid route cannot be found between 2 checkpoints (A and B), the RAM module is able to bypass checkpoint B and instead plan a route from checkpoint A to the checkpoint after B. This capability is in place to satisfy the fundamental ground rule: "Always complete the mission."

3.3 Camera Processing

A total of three different methods were tried to perform line detection for real time line detection from camera images. The setup is a Firewire 1394 camera from Point Grey systems, DragonFly2, interfaced via libdc1394 [9] to a "C" based Linux environment.

3.3.1 Design Choices

3.3.1.1 Method 1: Lane Detection From Polygons

This method extracts lane markings from colored polygon regions in the image. Color is inferred (*flatColorImage*) from the captured image (*newimage*) using HSV transform[10], and converted to a Grey scale image (*greyImage*). Edges (*edgeImage*) from the blurred image (*greyBlurImage*) were extracted using the Canny Edge [11], [12] algorithm and thinned using Non-Maxima suppression (*nmsImage*), which provided lines (*houghImage*) using Hough Transform [11]. Lines were trimmed, grouped (*groupedSegmentImage*) and averaged (*groupedSegmentAvgImage*) to generate polygons. The major pixel color within each polygon gave it the primary color. Adjacent polygons with same primary color were merged (*mergedPolyImage*). Lane markings were extracted from yellow and white Polygons.

3.3.1.2 Method 2: Lane Detection from Linear Parabolic Curves

As described in the paper [8], the road is modeled as a set of "near" Linear and "far" Parabolic curve segments. Capturing the image, extracting color and edge information is same as in previous method. The LAPACK [14] library was used to solve the 6x6 system $(A)^T W A C = (A)^T W B$, which gave the parameter values. These parameters were plotted as the Linear part of the curve (*LinearCurveImage*).

3.3.1.3 Method 3: Lane Detection from Color scan and Line Segments

This method is based on scanning the captured color image for yellow and white regions and extracting lines from such regions which were within reasonable limits of a typical

lane marking. This way the edge image was a skeleton of the yellow and white regions which resembled lane markings. Using this skeleton image, the lines and segments were determined as described in the first method, using thresholds tuned to give best results. At the end, this resulted in yellow and white lines perceived to be a good close approximation to the actual lane markings.

3.3.2 Camera Results

3.3.2.1 Method 1: Lane Detection From Polygons

The Figure 6 below has the images: Left to right, top to bottom: *newImage*, *greyBlurImage*, *edgeImage*, *nmsImage*, *groupedSegmentImage*, *groupedSegmentAvgImage*, *flatColorImage* (overlay *groupedSegmentAvgImage*), *mergedPolyImage.*

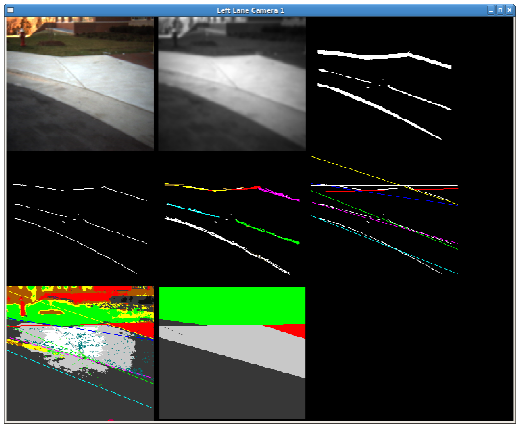

Figure 6 – Method 1

Figure 7 – Method 2

3.3.2.2 Method 2: Lane Detection from Linear Parabolic Curves

The Figure 7 above has the output of the linear solver in the bottom right image (*LinearCurveImage*).

3.3.2.3 Method 3: Lane Detection from Color scan and Line Segments

The Figure 8 below has the images: Left to right, top to bottom: *newImage*, *flatColorImage*, *edgeImage*, *groupedSegmentImage*, *groupedSegmentAvgImage*, *TrimmedLineSegments*, *TrimmedLineSegments*, *flatColorImage (with dark grey regions recolored as black)*, *LinearCurveImage.*

The observations from the 3 methods described above are summarized in Table 3. Method 3 is the preferred way to detect lane markings and was selected for use in our Urban Challenge vehicle.

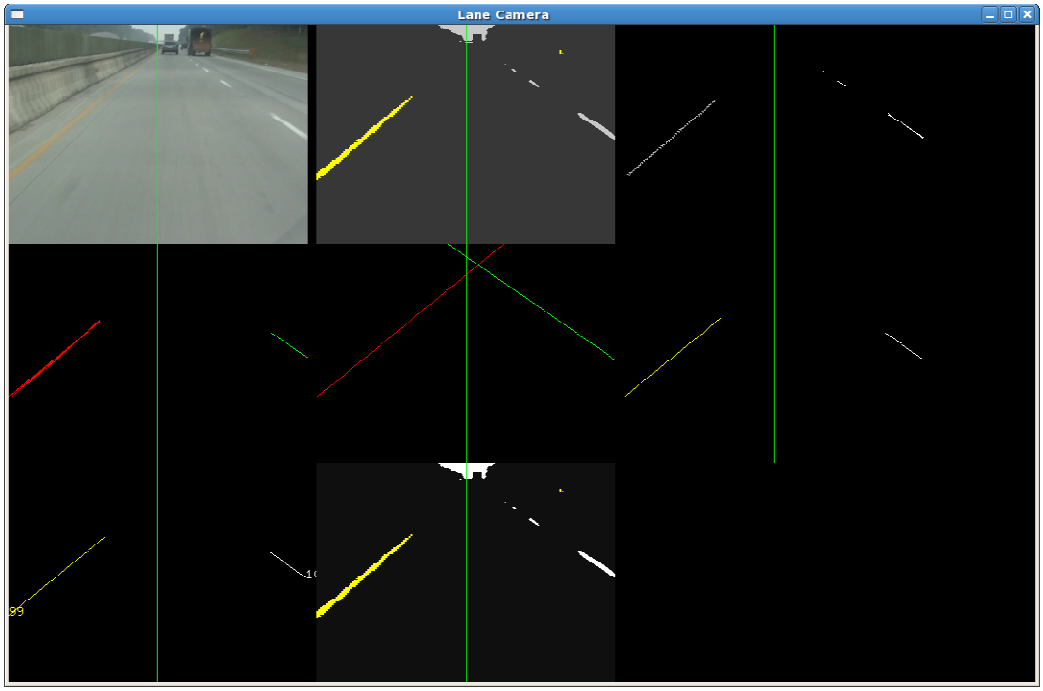

Figure 8 – Method 3

3.3.3 Observations and Analysis

The results from the three methods are compared in Table 3.

Feature	Method 1	Method 2	Method 3
Uses standard algorithms	Yes	Yes	Yes
Frame Rate achieved	Up to 5 fps	Up to 5 fps	Up to 10 fps
Main Errors introduced by	Merging non convex polygons	Imperfect edge detection and left/right split	Nothing major
Can detect multiple lanes	Might	No	Yes
Can detect solid vs dashed lane marking	No	No	Yes
Can detect yellow vs white lane marking	Yes	No	Yes
Good lane detection performance achieved only when	Small number of convex polygons	Proper edge detection and proper left/right edge pixels split	All cases

Table 3 – Comparison of Image Processing Methods

3.4 Lidar Processing

3.4.1 Analysis and Design

The vehicle uses 8 different SICK LMS 291lidars. They are located around the vehicle in a setup to gain 360 degrees of vision around the vehicle. They are used to find objects that are both above and below the road/ground surface. These units are designed to be used for close to midrange object detection. There are three major functions that are served by the sets of lidars. The first set detects objects above the ground, another set detects the contour of the surface ahead of and behind the vehicle, and the final set searches for edges of the driving surface.

The first two sets of lidars are used in conjunction with each other to create a more accurate representation of the surroundings. Their purpose is to find objects that are located above the current driving surface, for example other vehicles, sign posts, etc. These lidars are located on all four of the corners of the vehicle and on the center of the front and rear bumpers.

The four lidars on the corners are located at 45 degrees to the vehicle centerline and are looking outward on a horizontal plane. Since the angles do not allow for complete coverage around the vehicle, there are slight blind spots directly next to the side doors and directly behind the rear bumper. These blind spots are less than a meter in distance. These are not of concern because any object would have to travel through the field of at least one lidar before it would be able to enter a blind spot. The vision from each of the corner lidars also overlaps with at least one other unit. This is used to get better representations of the size and shape of the objects within the view. For example, the right and left front lidars could see different sides of an object independently; however, when their analysis is combined into one object we see a single object with a much more accurate shape.

The other lidars are located on the center of the front and rear bumpers of the vehicle. They are looking outward in a vertical direction. Their purpose is to build a contour map of the road surface in front of the vehicle. That is combined with the data from the four corner lidars to determine what portions of the points are at the same level as the ground. This helps us to clean up a picture and remove the ground from the data. This noise from the ground is seen in situations where the vehicle's suspension allows it to tilt towards the ground or when the ground slopes upward.

The third and final set of lidars on the vehicle is used to detect the edges of the driving surface. They are positioned on the front bumper and looking out vertically at a 45 degree angle. They scan outward from the vehicle and are looking for untraversable changes in terrain height. This change would indicate a ditch, curb, or edge of the road. Their main objective is to cause fine corrections to the vehicle to keep it off of the edges of the road.

The lidar information is consolidated and combined into objects that are described using shapes called convex hulls or convex envelopes [2]. A convex hull is the minimal convex set of points in real vector space that contains the union of a set of points. This is the idea that if a rubber band was wrapped around many points, it would be the shape that includes all of the points with only convex angles. These shapes have many advantages for this type of use. One main advantage is that it allows the software to minimize the number of points contained within an object for processing purposes while still keeping a valid shape and size. For example it would eliminate the inner points within a cluster/object. Another advantage is that it allows for a natural progression along the edges of obstacles. It stops the vehicle from becoming stuck within a concave object. For example, if there were an "L" shaped object in front of the vehicle, a concave object could allow the vehicle to drive into the inside corner of the L. However, since the object is convex, it would combine the two points on the edge of the L and therefore push the vehicle to avoid the inside of the L.

This implementation involves the use of the Graham-scan convex hull algorithm [2]. This algorithm was used due to its good mixture of simplicity in calculations and its efficiency. There are two main steps that are performed to complete this process. The first step is to sort the points based on their location within space. This process can be performed in O(n log n) time due to the nature of the sort used. Then once the points are all sorted the software iterates through the points three at a time. A calculation is performed to determine the angle each set of three points creates. If it is a concave angle then the center point is removed and the next point within the sorted list is added to the set of three points. Assuming the three points create a convex angle then the earliest point in the set of three is considered to be on the outer surface of the object. This process is considered O(n) because the sorted list is only passed through once. Therefore if we compare the two big O notations for each of the steps within the process, O(n log n) is the larger of the two time frames, therefore the entire process is performed in O(n log n) time.

3.4.2 Lidar Processing Results and Performance

The impressive performance of the lidar sensors is their extreme accuracy in their measurements; however, there are limitations to its use based on real world scenarios. The device documentation lists a maximum range of 80 m. However, we have determined through years of testing that the practical range of these sensors is roughly 45 to 55 m. At this range the values are quite consistent and very reliable. However, one more performance issue is the lack of reflectivity off of certain black objects. When this occurs, the lidar values where the black object is seen go to infinity, or state that there is actually nothing at that location. This issue is dealt with using the different sensors to make up for each one's shortcomings.

3.5 Testing

3.5.1 Methodology

Insight has a staged, systematic test methodology. The high level description of the Test Process is shown in Figure 9. The testing takes place in stages to isolate and test individual modules. This is followed by driver-in-the-loop testing for Basic Driving Behaviors, followed by testing with no driver. This process is again followed for the Advanced Driving Behaviors.

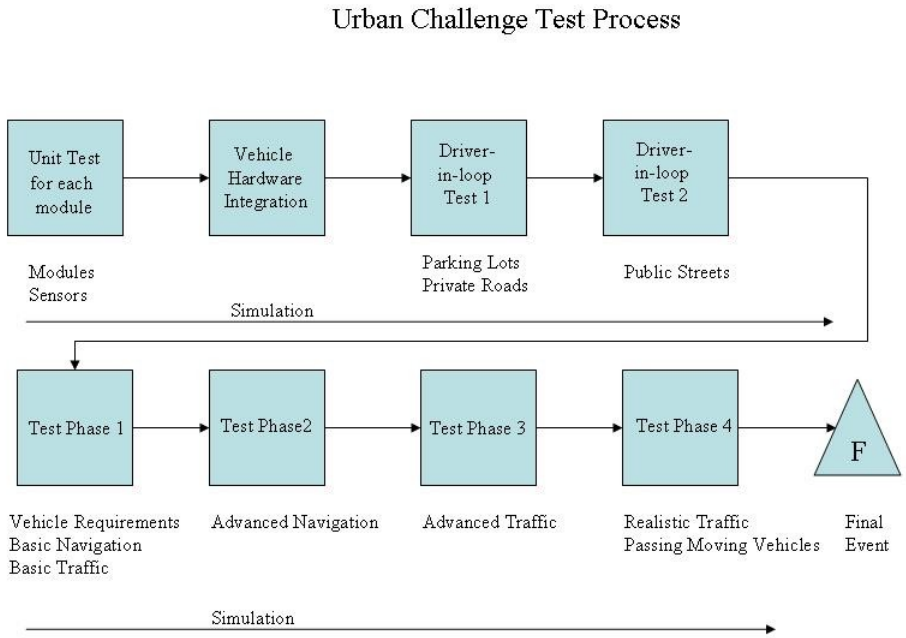

Figure 9 - Vehicle Test Process

3.5.2 Simulation

A key method that we used to debug and test our software was through the use of an intricate simulator. This software allowed us to gather information on actual missions around a defined course to replay later. We are able to replay missions with timing considerations in place to find out the vehicles reaction before actually running it on the car. This allows for more controlled testing of an uncontrolled environment. This allows us to be able to rerun the same information many times and view what difference our algorithm or parameter tuning causes.

4.0 Test Status

The effectiveness of the key modules was discussed previously, but this section shows the overall status of the project in its test phase. Table 4 shows the result of testing at the time of this paper.

TEST PHASE	STATUS	BEHAVIORS
Unit Test sensor modules	Completed	
Vehicle Hardware Integration	Completed	
Driver-in-loop Test 1	Completed	In parking lots
Driver-in-loop Test 2	Completed	Public Streets
Test Phase 1	Completed	Vehicle Requirements Basic Navigation Basic Traffic
Test Phase 2	In process	Advanced Navigation
Test Phase 3	In process	Advanced Traffic
Test Phase 4	In process	Realistic Traffic Passing Moving Vehicles

Table 4 – Testing Status

5.0 Summary

Insight Racing is now on its third generation of autonomous vehicle software. After a 12th place finish in the 2005 Grand Challenge, Insight has continued to develop and refine its architecture and technologies. This latest design is based on Commercial Off the Shelf (COTS) hardware and an updated system architecture.

Preliminary testing has confirmed the scalability and flexibility of this solution. After taking delivery of the Elise from Lotus, we were autonomously driving the vehicle within 4 weeks. Since then, we have continued to test the vehicle at different speeds and in different test environments.

The capability to do significant testing in a simulation mode has allowed for development and testing to continue even when weather conditions have not been suitable for testing or when the vehicle is unavailable. Simulation mode has also allowed us to test all new code without risk to the vehicle and test crew. A significant number of problems have been resolved without starting the vehicle.

Technical development has been targeted at improved information from camera technologies, better techniques for managing large numbers of physical objects, new decision making capabilities, and the ability to miniaturize our system. We continue to focus on reliability, safety, and backup systems to support safe operations of the vehicle at all times.

Insight Racing will continue to develop new and improved capabilities through the use of system simulation and active testing.

6.0 References

[1] Jaus Architecture for Unmanned Systems http://www.jauswg.org/
[2]*Computational Geometry In C (Second Edition)*, by O'Rourke, Cambridge University Press, 1998.
[3]*Handbook on Discrete and Computational Geometry*, by Goodman and O'Rourke (eds), CRC Press LLC, 1997.
[4] O'Rourke, J. *(1987).* Art Gallery Theorems and Algorithms. *Oxford University Press.* ISBN 0-19-503965-3.
[5] E. W. Dijkstra: *A note on two problems in connexion with graphs*. In: *Numerische Mathematik*. 1 (1959), S. 269–271
[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.3: Dijkstra's algorithm, pp.595–601.
[7] *Urban Challenge Route Network Definition File (RNDF) and Mission Data File (MDF) Formats*, Defense Advanced Research Projects Agency, Arlington, VA, March 14, 2007
[8] An Improved Linear-Parabolic Model for Lane Following and Curve Detection: Claudio Rosito Jung and Christian Roberto Kelber. http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/10657/33624/01599093.pdf?arnumber=1599093
[9] libdc1394: http://damien.douxchamps.net/ieee1394/libdc1394/index.php
[10] HSV Color Space: http://en.wikipedia.org/wiki/HSV_color_space
[11] Machine Vision: Wesley E. Snyder, Hairong Qi (ISBN: 052183046X)
[12] Canny Edge Detection Tutorial: http://www.pages.drexel.edu/~weg22/can_tut.html
[13] OpenGL: http://www.opengl.org/
[14] LAPACK: http://www.netlib.org/lapack/
[15] *Sample Route Network Definition File (RNDF) Rev 1.5* , Defense Advanced Research Projects Agency, Arlington, VA, March 29, 2007